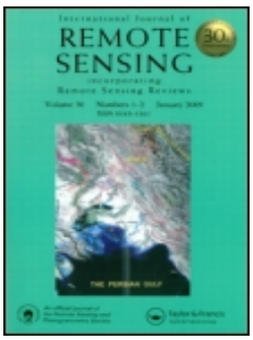

International Journal of Remote Sensing

ISSN: 0143-1161 (Print) 1366-5901 (Online) Journal homepage: http://www.tandfonline.com/loi/tres20

Using point cloud data to identify, trace, and regularize the outlines of buildings

M. Awrangjeb

To cite this article: M. Awrangjeb (2016) Using point cloud data to identify, trace, and regularize the outlines of buildings, International Journal of Remote Sensing, 37:3, 551-579, DOI: 10.1080/01431161.2015.1131868

To link to this article: http://dx.doi.org/10.1080/01431161.2015.1131868

Published online: 21 Jan 2016.

INTERNATIONAL JOURNAL OF REMOTE SENSING, 2016
VOL. 37, NO. 3, 551–579
http://dx.doi.org/10.1080/01431161.2015.1131868

Taylor & Francis
Taylor & Francis Group

Using point cloud data to identify, trace, and regularize the outlines of buildings

M. Awrangjeb

School of Engineering and Information Technology, Federation University Australia, Churchill Vic, Australia

ABSTRACT

Rectilinear building outline generation from the point set of a building usually works in three steps. Boundary edges that constitute the building outline are first identified. A sequence of points is then traced from the edges to define the building boundary. Finally, lines are generated from the sequence of points and adjusted to form a regular building outline. Existing solutions have shortcomings in one or more of the following cases: identifying details along a concave shape, separate identification of a 'hole' inside the shape, proper boundary tracing, and preservation of detailed information along a regularized building outline. This article proposes new solutions to all three steps. By using the maximum point-to-point distance in the input data, the solution to the identification step properly detects the boundary edges for any type of shape and separately recognizes holes, if any, inside the shape. The proposed tracing algorithm divides boundary edges into segments, accurately obtains the sequence of points for each segment and then merges them, if necessary, to produce a single boundary for each shape. The regularization step proposes an improved corner and line extraction algorithm and adjusts the extracted lines with respect to the automatically determined principal directions of buildings. In order to evaluate the performance, an evaluation system that makes corner correspondences between an extracted building outline and its reference outline is also proposed. Experimental results show that the proposed solutions can preserve detail along the building boundary and offer high pixel-based completeness and geometric accuracy, even in low-density input data.

ARTICLE HISTORY
Received 11 May 2015
Accepted 9 December 2015

1. Introduction

Automatic building extraction is important for various real-world applications such as urban modelling, disaster management, and homeland security. The complete solution to building extraction from light detection and ranging (lidar) point cloud data has two parts. Part 1 essentially involves a segmentation technique where each building is identified as a subset of the input point set (Part 1: Segmentation). This segmentation technique separates the non-ground points (mainly buildings and trees) from the ground points and removes the trees from the non-ground points (Awrangjeb and

CONTACT M. Awrangjeb Mohammad.Awrangjeb@federation.edu.au

Fraser 2014b). Part 2 generates the individual building outline from its point set (Part 2: Outline Extraction). Boundary edges that form the boundary are first identified. Each edge is a line that connects two consecutive boundary points. They are then traced to form an irregular building boundary. Finally, the irregular boundary is regularized in order to form a regular outline. The investigation reported in this article concentrates on Part 2 of the building extraction process, namely building outline extraction.

Methods of building outline extraction from point cloud data fall into two major categories: intensity- and contour-based. In the intensity-based approach, corners and lines representing the regular building outline polygon are extracted after converting the input point set into an intensity image (Hinks et al. 2009; Grigillo and Kanjir 2012). This approach inherently combines the boundary edge identification, tracing, and regularization steps (Laefer et al. 2011).

In the contour-based approach, a contour (an irregular polygon) representing the exact or an approximate boundary is extracted from the input point set (Awrangjeb and Fraser 2014a) (boundary edge identification and tracing). The extracted contour consists of a series of points, which may be represented by points from the input point set (Zhou and Neumann 2013). In many cases, the extracted contour is then regularized and divided into line segments to obtain a regular polygonal building boundary (Sampath and Shan 2007). Many authors (Alharthy and Bethel 2002; Maas and Vosselman 1999) have concentrated on boundary regularization, rather than on how the boundary contour is extracted.

The boundary regularization methods in the literature can also be divided into two approaches: model-driven (Kwak and Habib 2014) and data-driven (Sampath and Shan 2007). The model-driven approach tries to fit certain shapes to the data, while the data-driven approach tries to extract shapes present in the data. Although the model-driven methods are robust, their performance is limited to known models. The data-driven methods work, in general, for any rectilinear building shapes.

In this article, a contour-based approach to the identification, delineation, and regularization of building boundaries from lidar point cloud data is presented. For a given point cloud data, the proposed boundary identification technique first forms an initial building boundary (in terms of edges) using the outside edges of the Delaunay triangulation. Then, in order to obtain the final boundary, it gradually replaces the initial edges with the inside edges, which reside close to the current boundary. The tracing algorithm first decomposes the boundary edges into one or more edge segments and forms a tree structure. Then, it finds the largest edge segment in the tree structure as a series of points. The regularization of the extracted contour follows a data-driven approach based on straight lines, which are extracted using a newly proposed improved corner and line extraction algorithm. The principal directions of the building are automatically determined and other lines are adjusted. The regular building boundary, which will be referred to simply as the building footprint in this article, is at last produced by taking the intersection points of adjusted lines. In order to assess the quality of the extracted boundary, a new objective evaluation system is proposed using the distance and parallelism of corresponding corners and lines of the reference and extracted building outlines. The proposed evaluation technique evaluates the extracted irregular boundaries as well as the regular extracted lines. In order to avoid any bias, that may have been introduced in the point cloud segmentation (Part 1: Segmentation), the point

set inside each reference building is directly provided as input to the proposed building outline generation (Part 2: Outline Extraction).

2. Related work

Building boundary extraction methods in the contour-based category can be further categorized into two groups: indirect and direct contour extraction. For boundary regularization, straight lines are first extracted along the extracted boundary and then adjusted in dominant building directions.

2.1. *Indirect contour extraction*

The input point set of a building is first used to generate a binary image (again represented by a grid of cells), where the background represents the ground and the foreground represents the building (Galvanin and Dal Poz 2012; Yang, Xu, and Dong 2013). The grid or image may be aligned with an estimated dominant direction of the building (Alharthy and Bethel 2002; Verma, Kumar, and Hsu 2006). Then, the boundary is extracted using an edge detector around the foreground (Haithcoat, Song, and Hipple 2001; Awrangjeb and Fraser 2014a), or the boundary of the foreground cells is directly accepted as the building footprint (Verma, Kumar, and Hsu 2006). Sometimes, the input points close to the extracted boundary replace the edge points (Awrangjeb and Fraser 2014a).

Awrangjeb and Fraser (2014a) converted the point cloud into a binary mask (0.25 m resolution) and used the Canny edge around the black shape in the mask as the building boundary. In a second method, they used a mask resolution of 1 m and replaced the Canny edge points by the nearest lidar points. Haithcoat, Song, and Hipple (2001) used a Gaussian kernel to extract building boundaries from a normalized digital surface model of 1.3 m resolution. Verma, Kumar, and Hsu (2006) used a grid of 2 m resolution and aligned the grid with the dominant direction of the building. The grid cells containing enough input points were marked and the grid lines of the marked cells were considered to delineate an approximate building boundary. A similar method was proposed in Zhou and Neumann (2013), where a boundary grid line of a marked cell was replaced by the nearest input point. Nonetheless, similar to intensity-based methods, the indirect contour extraction methods suffer from the problem of appropriate image resolution selection. Moreover, some of the input points may reside outside the extracted contour (Awrangjeb and Fraser 2014a; Zhou and Neumann 2013).

2.2. *Direct contour extraction*

The boundary contour is directly extracted from the input point set, as in the convex hull algorithm (De Berg et al. 2000). However, the convex hull algorithm is not suitable for extracting concave outlines (Sampath and Shan 2007). Jarvis (1977) modified the convex hull formation algorithm by limiting the search space using a circular neighbourhood, which allowed some concavities in the extracted shape. However, this algorithm was not found to be particularly useful due to an uneven point distribution in the point cloud (Sampath and Shan 2007). In turn, the circular neighbourhood was replaced with a

rectangular neighbourhood, whose dimensions were set at least twice the point spacing in and across the scan direction (Sampath and Shan 2007). Lahamy (2008) used this modified version for building footprint extraction. Wang and Shan (2009) proposed a further modified algorithm, which iteratively removed the non-boundary points by a local convex hull test. This algorithm was found sensitive to both the neighbourhood and the distance parameters (Wang and Shan 2009).

The main problem with the above algorithms is that an inner boundary ('hole' inside a shape) of a given point set cannot be extracted. Edelsbrunner, Kirkpatrick, and Seidel (1983) proposed the so-called α-shape determination algorithm that is able to extract both the inner and the outer boundaries of a shape. Dorninger and Pfeifer (2008) used an α value which was twice the mean point-to-point distance in the input point set. The non-parametric approach in Peethambaran and Muthuganapathy (2015) is dependent on the homogeneity of the input data and thus does not perform well in random point cloud data. Researchers have used the minimum spanning tree and/or a depth-first search algorithm to obtain the boundary edge sequence from the discrete edges (Wang and Shan 2009).

2.3. *Boundary regularization*

The Douglas–Peucker (DP) algorithm (Douglas and Peucker 1973) has been extensively used to approximate the extracted building contour with a number of straight lines (Jwa et al. 2008). However, the extracted lines are visually unattractive as they are not necessarily parallel or perpendicular to each other for rectilinear building boundaries. In addition, the DP algorithm may remove critical points depending on the height threshold (Zhang, Yan, and Chen 2006). Weidner and Förstner (1995) applied an optimization algorithm based on the local minimum description length (MDL) to the extracted DP lines to generate 10 different orthogonal models. However, this method was sensitive to the initial point selection because of its local MDL approach and it generated oversimplified footprints (Jwa et al. 2008). Jwa et al. (2008) proposed a global optimization approach, where the optimum solution was achieved when a building footprint was maximally hypothesized as the repetition of identical line slope.

Many authors simplified the regularization problem by limiting the building walls to be in two perpendicular directions. Alharthy and Bethel (2002) used an orientation histogram approach to determine the two dominant wall directions. All boundary points were then connected by lines to obtain an approximate building footprint, where each line was adjusted into one of these two directions. Sampath and Shan (2007) followed a similar approach by grouping the directions of long boundary lines (more than 10 m) into 'horizontal' and 'vertical' directions. A similar method was proposed in Zhou and Neumann (2013) where instead of two, a set of dominant directions were determined from a large local area, say 1 km $\times$ 1 km, using the same histogram approach as in Alharthy and Bethel (2002).

Maas and Vosselman (1999) proposed an approximation based on the main building orientation obtained from the directions of the horizontal intersection lines between the roof faces. Obviously, this regularization would not work for buildings having only flat roof segments (i.e. high-rise buildings), since such a building does not have a horizontal intersection line that could be considered as the main building orientation. Similar

methods were proposed in Dorninger and Pfeifer (2008) and Perera, Nalani, and Maas (2012), where the longest boundary line was considered as the main building orientation. Zhang, Yan, and Chen (2006) estimated the dominant directions based on the assumption that the total number of orthogonal segments is higher than the total number of oblique segments in the building boundary. Then, four operations, split, merge, intersect, and remove, were applied to align the segments to the dominant directions.

3. Proposed boundary extraction

For a given point cloud P of a building, the proposed boundary extraction algorithm works in two major steps: boundary identification and boundary tracking. In the first step, a boundary connecting the points along the periphery of P is identified as an unordered set of edges. The line connecting any two consecutive boundary points is called an edge. A variation of the first step is also presented in order to extract 'holes' or 'concavities' inside some shapes, for example, the A-shape. In the second step, edges in the unordered set are tracked in order to find an ordered set of points or boundary. If there are two or more disconnected objects in P, the algorithm finds one boundary for each individual object. The proposed boundary extraction algorithm is described below using a number of shapes, including the C-shape shown in Figure 1(*a*).

The 'hole' extraction method based on long edges in the triangulation was previously employed for facade feature extraction (Pu and Vosselman 2007; Truong-Hong et al. 2012). Although the proposed boundary extraction method was independently developed, it is a technical redesign of the existing hole extraction method for generation of

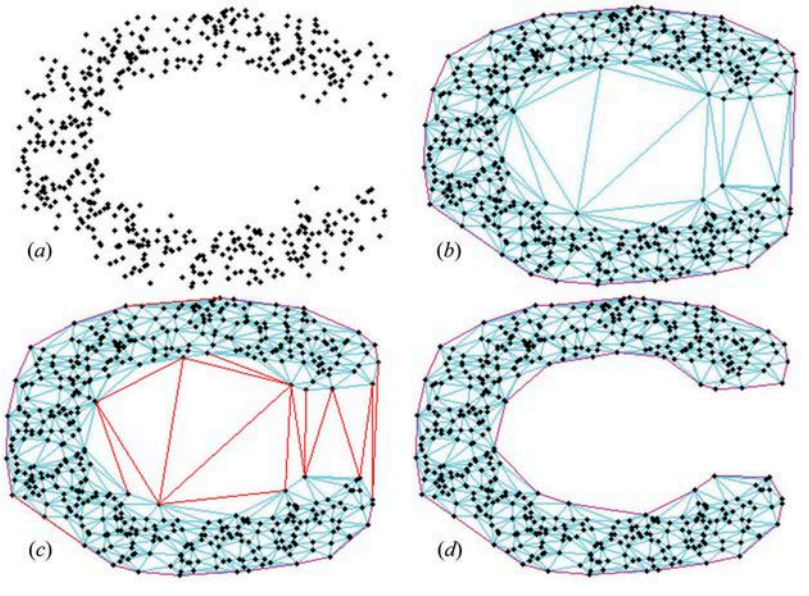

Figure 1. Boundary extraction from the C-shape: (*a*) input point cloud P, (*b*) Delaunay triangulation and the initial boundary (magenta colour), (*c*) removal of long boundary edges that are shown in red colour, and (*d*) boundary in magenta after the removal of long edges.

building footprint, particularly for boundary identification as well as for hole extraction. However, the extraction of the boundary of a large point set from the boundaries of two or more of its point subsets is completely new. Any inside hole can also be determined from the boundaries of subsets.

3.1. *Boundary identification*

A Delaunay triangulation is formed for P. As shown in Figure 1(*b*), every side of a triangle inside the triangulation is associated with exactly two neighbouring triangles. However, for a triangle along the periphery of P, one of the sides is associated with only one triangle. All such edges form the initial boundary, shown in magenta colour in Figure 1(*b*).

Some of the edges in the initial boundary connect the boundary points that are far away from each other, that is, they connect non-neighbouring points. Such edges should be removed to reveal any concavities present in the shape or if two or more objects in P are to be disconnected in order to obtain individual boundaries. Let $d_{\max}$ indicate the maximum point-to-point distance in P, that is, there is at least one neighbouring point $q \in P$ within $d_{\max}$ of a point $p \in P$. If an initial boundary edge is longer than $2d_{\max}$ (threshold $T_d = 2d_{\max}$), then it is removed. This removal leaves each of the two other sides of the same triangle to be now associated with only one triangle. Consequently, the two other sides now become two new boundary edges and are separately tested if each of them connects two neighbouring points. The removal of long boundary edges iteratively continues until every edge along the boundary is at most T_d in length. Figure 1(*c*) shows the edges to be removed in red and Figure 1(*d*) shows the final boundary in magenta after the removal of long edges. Note that it is assumed that like in many existing methods (Awrangjeb and Fraser 2014b), the value of $d_{\max}$ is available as an input in the proposed method. If not, it can be estimated using an analysis based on either the point density (Sampath and Shan 2007) or the edge-length histogram (Truong-Hong et al. 2012).

In order to find an inner cavity within a shape, for example, the hole of the A-shape in Figure 2(*a*), the same procedure can be applied. After obtaining the boundary edges (see Figure 2(*b*)) using the procedure discussed above, the boundary edges are not considered anymore, but all the edges inside the boundary are tested if they are within one or more potential holes. If an inside edge longer than T_d is found, then it is removed and the remaining four sides of the two associated triangles on both the sides are further tested. Consequently, the iterative procedure removes all the long edges as shown by red lines in Figure 2(*c*). Figure 2(*d*) shows the boundary edges of the hole after the removal of long edges.

3.2. *Boundary from point subsets*

An important property of the proposed boundary extraction algorithm is that it can extract the boundary of a large point set P from the boundaries of two or more subsets of P. Any hole inside P can also be obtained separately from the boundaries of subsets. This is particularly useful for obtaining a building boundary from the boundaries of its

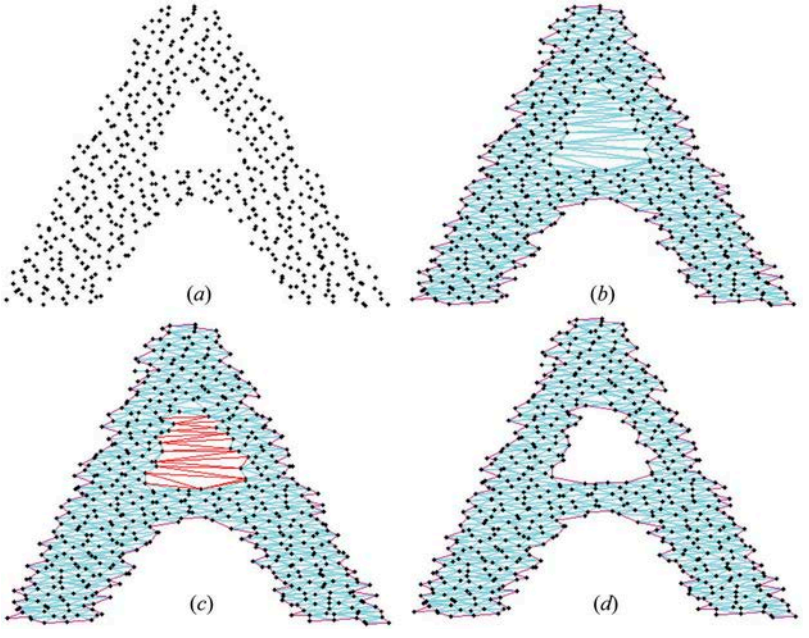

Figure 2. Inside hole extraction for the A-shape: (*a*) input point cloud P, (*b*) outside boundary, (*c*) inside long edges shown in red, and (*d*) hole in magenta colour after the removal of long edges.

planes (Awrangjeb, Lu, and Fraser 2014). The computational cost is also reduced since now the Delaunay triangulation is formed on small subsets of P.

Let the A-shape is decomposed into four point subsets as shown in Figure 3(*a*). The subsets can be simply obtained from the original point set shown in Figure 2 by using four polygons that pass through the hole inside the shape. The four boundaries of the subsets are shown in Figure 3(*b*). In order to find the outside boundary of the A-shape, only the points on these four boundaries are now given as the input to the boundary extraction algorithm. Let the input point set be $P_s \subset P$. There is no change required in the procedure discussed in Section 3.1, since the algorithm for the extraction of the outside boundary works only on the outside edges. Figure 3(*c*) shows the resulting outside boundary in pink colour and the inside edges in cyan colour.

Let the set of inside edges in Figure 3(*c*) be E_i. If there is a hole inside a point subset, that hole can be extracted without any modification to the hole identification algorithm discussed above. However, if there is a hole among the subset boundaries, as in Figure 3, the algorithm for obtaining the inside hole from the original point set can be applied with the following modification, which is necessary in order to keep the subset boundaries unchanged as well as to effectively extract the hole among the subset boundaries.

From E_i, those edges that reside inside any of the subset boundaries are removed. Figure 3(*d*) shows the remaining inside edges in yellow colour and the subset boundaries in four different colours. The hole extraction algorithm is now applied to the remaining edges. Figure 3(*e*) shows the edges in red colour that have been removed. As with the original algorithm, the modified hole extraction algorithm does not remove the edges that are already on any of the subset boundaries or on the extracted outside boundary. Figure 3(*f*) shows the inside as well as the outside boundaries.

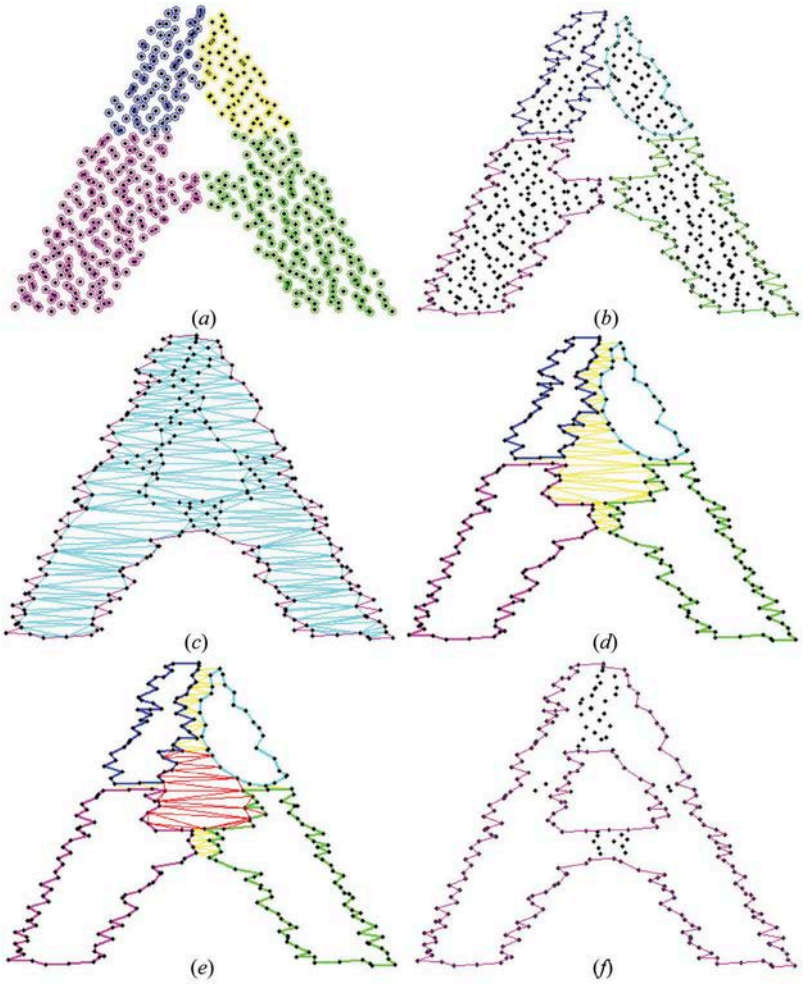

Figure 3. Boundary generated from point subsets: (*a*) input point cloud P decomposed into four subsets, (*b*) four boundaries of point subsets, (*c*) outside boundary of P obtained from its four subset boundaries, (*d*) inside edges within the gap of subset boundaries in yellow colour, (*e*) removed inside edges in red colour, and (*f*) final hole and outside boundaries.

Figure 4(*a*) shows the point cloud in red colour for a practical building scene ($d_{max} = 0.2$ m). Figure 4(*b*) shows the extracted plane boundaries by a lidar segmentation algorithm in Awrangjeb and Fraser (2014b). Although there are 6006 lidar points in the point cloud in Figure 4(*a*), there are only 1161 points on the nine plane boundaries in Figure 4(*b*). Figure 4(*c*) shows the extracted roof boundary from the plane boundaries.

3.3. *Boundary tracing*

Once the boundary of the input point cloud P is identified, it is tracked before regularization. For each boundary, the proposed tracking algorithm first constructs a tree structure consisting of one or more edge segments and then looks for loops of edge

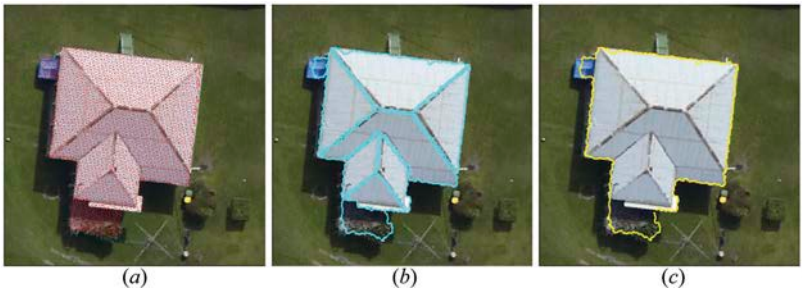

Figure 4. A sample building scene taken from a test data set: (*a*) point cloud in red colour, (*b*) individual plane boundaries, and (*c*) roof boundary obtained from the plane boundaries (overlaid on the orthophoto).

segments within the tree. Each loop is represented by a sequence of vertices. A single sequence of vertices is finally obtained for each boundary through identifying the super loop that contains all the points of the boundary.

The tracking algorithm is described here through a small but real point cloud P in Figure 5(*a*), where the original triangulation is shown along with the initial boundary. The boundary identification algorithm in Section 3.1 only removes the outside edges that are longer than the threshold. So, any long edges that reside inside a boundary or short edges that may exist outside a boundary still remain. Figure 5(*b*) shows all the surviving edges after application of the boundary identification algorithm. There are three types of edges: pink (boundary), cyan (inner), and green (unexpected). As can be seen, the green and cyan edges are not associated with any boundary, thus the proposed boundary tracking algorithm only looks for the sequence of pink edges. Consequently, the inner and unexpected edges are automatically ignored by the proposed boundary extraction technique. While tracing boundary segments, each pink edge $E(i,j)$, between Vertices i and j, is visited only once, but a vertex (e.g. Vertex 13 in Figure 5(*b*)) in a junction of loops may be visited more than once.

Clearly, there are two object boundaries in Figure 5(*b*). While the top boundary consists of a simple loop, the bottom boundary contains two loops. For each boundary, the proposed algorithm starts visiting a vertex from a randomly selected unvisited edge, creates a new segment using the two vertices of this edge, and continues accumulating vertices into the segment through visiting neighbouring unvisited edges until it reaches the start vertex or a loop junction. For a simple loop boundary, it eventually reaches the start vertex and stops accumulating vertices into the current segment. However, for a complex boundary that has two or more loops, when the algorithm reaches at a junction, the growth of the current segment may continue through its one or more child segments. At a junction, there are two or more unvisited edges. The unvisited edge that makes the smallest clockwise angle with the previously visited edge (through which the junction is reached) is used to generate a child segment. Once the growth of the child segment stops, other unvisited edges from the junction will be used to generate successive child segments.

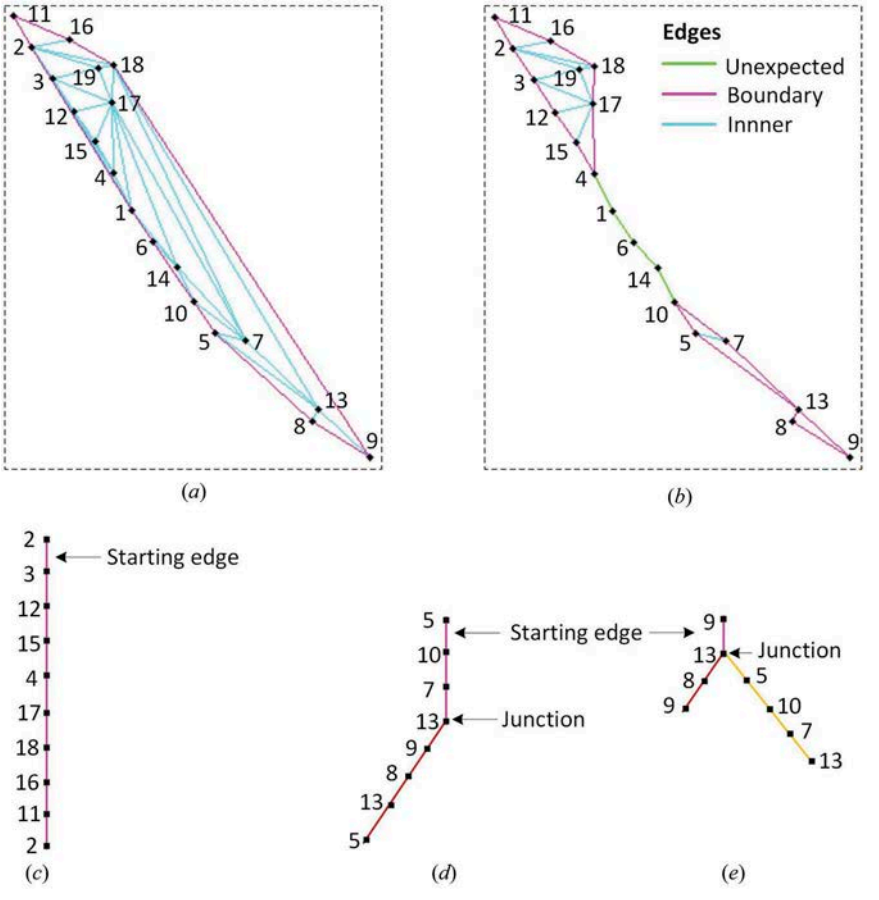

Figure 5. Boundary tracking: (*a*) input point cloud P (labelled with numbers 1–19) and initial boundary (magenta colour), (*b*) final two boundaries (magenta colour), edge segments for (*c*) top-left and (*d*) bottom-right boundaries in (*b*), and (*e*) alternative segments for bottom-right boundary in (*b*).

For example, Figure 5(*c*) shows the segment for the top boundary in Figure 5(*b*) when the algorithm starts from edge $E(2, 3)$. For a simple loop boundary, the algorithm finds the same segment independent of the start vertex. Figures 5(*d*) and (*e*) show two possible sets of edge segments, depending on the start edge, for the bottom complex boundary in Figure 5(*b*).

Individual object boundary can now be obtained by determining its super loop. For a simple boundary, the only loop, which is also the super loop, can be easily found by looking at the first and last vertices of the generated edge segment (Figure 5(*c*)). For a complex boundary, two or more loops are obtained by looking at the repeating vertices in its tree structure. In Figure 5(*d*), there are two loops starting with Vertices 5 and 13. The loop with the start Vertex 5 is the super loop and the other loop is discarded. In Figure 5(*e*), there are two loops starting with Vertices 9 and 13. In order to find the super loop, both loops are merged.

4. Proposed boundary regularization

Figure 6(*a*) shows the extracted roof boundary for another practical building scene ($d_{max} = 0.2$ m). The extracted boundary did not align well with the building boundary in the image because of the registration error between two data sources. As can be seen, there is a significant irregularity or noise in the lidar-derived building boundary. Thus, the first step of the proposed regularization method is to smooth the extracted boundary and to find the corners in order to decompose the boundary into lines. The contour-based corner detectors work fine with edges from regular grids, for example, edges from an image. However, they are quite ineffective with edges derived from point cloud data due to the random nature of the boundary points. Consequently, an improved contour-based corner detector is proposed in order to extract corners and lines. The lines are then adjusted assuming their orthogonal relationship with the neighbouring long lines.

4.1. *Corner detection and line extraction*

Figure 7 shows the flow diagram for the proposed corner and line extraction procedure from a given building boundary. It has two main parts: initial corner detection and line

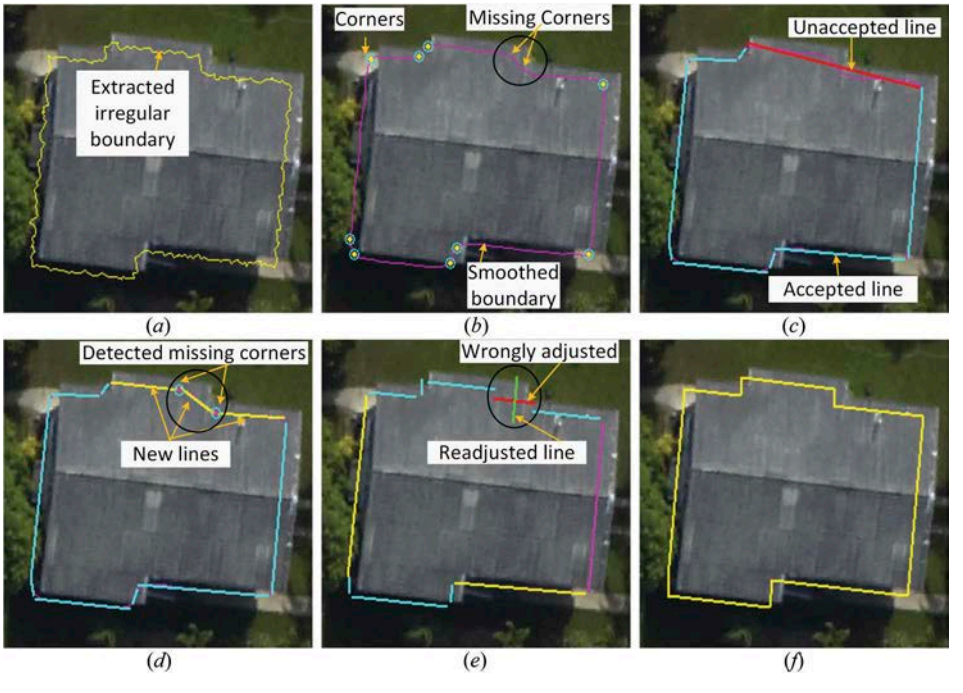

Figure 6. Regularization of a building boundary (lidar-derived boundaries are superimposed on an image which is not a true orthophoto, thus a significant misalignment is noticed between each boundary and its corresponding building in the image): (*a*) extracted irregular boundary, (*b*) smoothed boundary and corners, (*c*) extracted lines in cyan colour and an unaccepted line in red colour, (*d*) detected missing corners and more extracted lines, (*e*) adjusted lines, and (*f*) regularized building footprint.

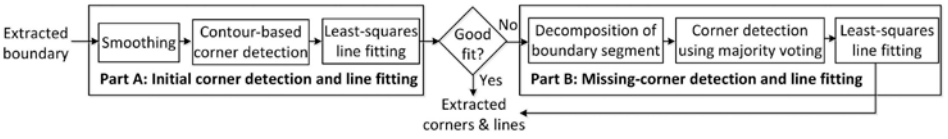

Figure 7. Flow diagram for corner and line extraction from a building boundary.

fitting (Part A) and missing-corner detection and line fitting (Part B). The output of the procedure consists of the extracted corners and lines.

In initial corner detection and line fitting (Part A in Figure 7), the contour-based corner detector in Awrangjeb, Lu, and Fraser (2012) is used. A Gaussian smoothing with a scale of $\sigma = 3$ is first applied to the extracted boundary in order to reduce the random nature of the point cloud data. Then, curvature peaks along the smoothed boundary are obtained as corners. The extracted corners divide the input boundary into segments. Figure 6(*b*) shows the smoothed boundary in magenta and detected corners in yellow dots within cyan circles. The boundary points between two successive corners constitute a curve segment.

Assuming that a building side is at least $d_m = 1$ m long, a straight line is then fit to each of the long curve segments using a least-squares technique. In order to find a good fitted line, the line is rotated following an iterative procedure to minimize the mean perpendicular distance d_{mean} of points to the line. If the minimum d_{mean} of a line is lower than T_d, the line is accepted as an appropriate line along the building boundary. If d_{mean} is higher than T_d, the line is not accepted. This can be due to missing of one or more corner points within the corresponding curve segment. Figure 6(*b*) shows all the accepted lines in cyan and an unaccepted line in red.

In order to find missing corners and, thereby, fit more lines (Part B in Figure 7), the following procedure is applied to each unused (where no line has been fit yet) curve segment, if any, between two accepted lines. Figure 8(*a*) shows an unused curve segment E_1E_2 from Figure 6(*c*) where a line was not accepted. A distance function f_d, shown in Figure 8(*b*), is first calculated for E_1E_2 by taking (signed) perpendicular distances from curve points to the line connecting two ends E_1 and E_2. f_d is then divided into segments by considering its zero crossings. A zero crossing is a point in the curve segment where f_d changes signs. For example, Figure 8(*b*) shows two zero crossings for E_1E_2 in Figure 8(*a*), where f_d changes signs from positive to negative and negative to positive, respectively. So, f_d as well as E_1E_2 are decomposed into three segments as shown in Figure 8(*b*).

For each segment, the minimum and maximum distance points are now obtained along the vertical axis from the absolute form of its corresponding distance function. Figure 8(*c*) shows that there are two maxima (M_1 and M_2) and three minima (m_1, m_2, and m_3) points for the absolute distance function of Segment 1 from Figure 8(*b*). Figure 8(*d*) shows that there is one maximum (M_3) and two minima (m_4 and m_5) points for the absolute distance function of Segment 2 from Figure 8(*b*).

For each maximum to be accepted as a corner point, it should have a distance difference of at least d_{max} in vertical direction and at least 1 m in horizontal direction from each of its two neighbouring minima points. While the vertical distance threshold

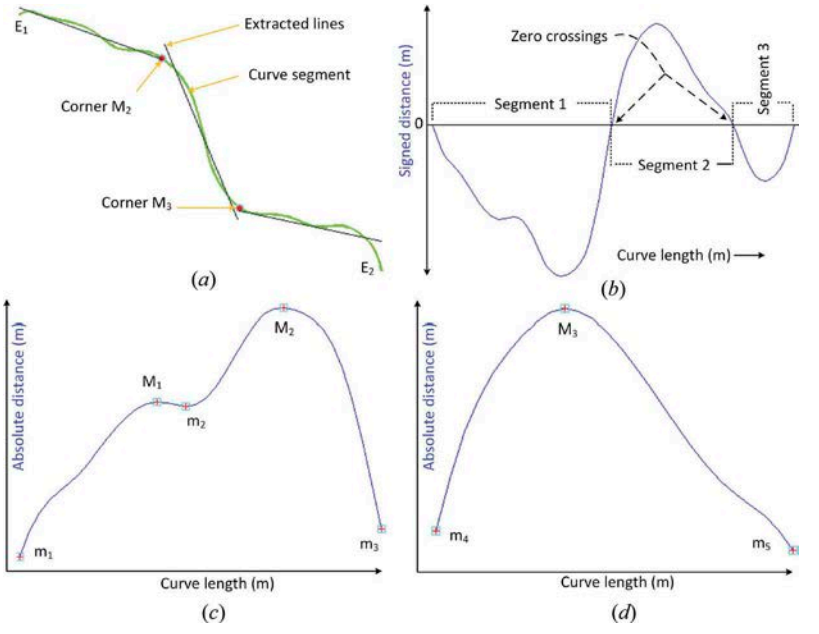

Figure 8. Extraction of missing corners and lines from a curve segment: (*a*) curve segment with newly extracted corners and lines, (*b*) decomposition of the distance function of the whole curve segment in (*a*) into three segments, and (*c*, *d*) maximum (M_1 to M_3) and minimum (m_1 to m_5) points for Segments 1 and 2 in (*b*).

interprets the point density in the input point cloud, the horizontal distance threshold is based on the assumption that the minimum length of a roof side is $d_m = 1$ m.

The missing corners are now effectively detected through a voting system, where one vote is allocated to a maximum point when it satisfies each of the above four conditions (vertical and horizontal conditions with its two neighbouring minima points). Initially, only those maxima points that get the maximum four votes are accepted as corners. Then, if there are two or more maxima points having less than four votes remain, the one that has the lowest vote is discarded and its neighbouring unaccepted maxima points are re-voted after changing their minima points. After the re-voting, again the maxima points that get the maximum four votes are accepted as corners and if there are two or more maxima points having less than four votes still remain the vote is casted again after discarding the maximum point that has the lowest vote. Thus, more corners are iteratively accepted from the remaining maxima points.

For example, from Figure 8(*c*), at the beginning M_1 gets two votes and M_2 gets three votes. So, M_1 is removed and a neighbouring minimum of M_2 is changed from m_2 to m_1, resulting in a total of four votes for M_2. Eventually, one corner is found for each of Segments 1 (Figure 8(*c*)) and 2 (Figure 8(*d*)), but no corner is found for Segment 3 in Figure 8(*b*).

Figures 6(*d*) and 8(*a*) show the two newly detected corners. These two corners decompose the curve segment E_1E_2 into three parts, and thus three lines are newly extracted using the same procedure discussed above. The new lines are shown in yellow in Figure 6(*d*) and in black in Figure 8(*a*).

4.2. *Line adjustment and footprint generation*

Once the extracted boundary is decomposed into several lines, they are now adjusted based on the long lines in order to generate a rectilinear building footprint. The adjustment procedure works in two steps: determination of principal or dominant directions and adjustment of lines, which are described with the help of lines shown in Figure 6(*d*).

In order to determine the principal directions, long lines that are at least twice the minimum building width $W_{\min}$ in length ($W_{\min} = 3$ m; Awrangjeb, Ravanbakhsh, and Fraser (2010)) are obtained. However, there may be small buildings which may not have such a long side. So, it is assumed that most of the buildings have only one principal direction and only big buildings may have more than one principal direction. Buildings that have two or more principal directions, the length of a building side along the second or a latter principal direction is at least 6 m long. For an extracted building boundary, if at least one long line is not found, the length-threshold is decreased by 1 m at each step until at least one long line is found. The long line L_f that has the least error d_{mean} is kept fixed and all other long lines, if any, are adjusted by making them parallel or perpendicular to the fixed line. For adjustment, the angle δ between L_f and a long line L_a is estimated and if $\delta \leq T_\delta$ (for parallel) or $\frac{\pi}{2} - \delta \leq T_\delta$ (for perpendicular), where $T_\delta = \frac{\pi}{16}$, L_a is adjusted. For example, the long magenta colour line in Figure 6(*e*) is kept fixed and the two yellow colour long lines are made exactly parallel or perpendicular to this fixed line. Usually, a building has one principal direction. However, if $\delta > T_\delta$ for an extracted long line, it is determined that the building being delineated has a second principal direction. More principal directions for large buildings can thus be determined and added. All other long lines that are found parallel or perpendicular to a principal direction are adjusted as discussed above. The short lines are then adjusted to their neighbouring long lines using $T_\delta = \frac{\pi}{4}$, since extracted short lines that may be as small as 0.5 m in length may be arbitrarily oriented due to the lack of enough curve points. Figure 6(*e*) shows all the adjusted lines.

There may be some short lines misadjusted. As shown within the black circle in Figure 6(*e*), the red colour line was adjusted wrongly as the corresponding extracted line shown in Figure 6(*d*) is almost parallel to its both neighbours. This line is from a short building side, whose detail is lost by the smoothing operation to the corresponding curve segment shown in Figure 6(*b*). In order to avoid such a wrong adjustment, if there is a short line L_s for which $\delta > \frac{\pi}{6}$ and two of its neighbours are parallel to each other, L_s is readjusted to make it perpendicular to its neighbouring lines. The green colour line shown within the black circle in Figure 6(*e*) is the readjusted line.

Finally, in order to find a building footprint, a perpendicular line is inserted in between two successive parallel lines, if any. The intersection points are determined for consecutive lines in order to obtain a regularized building footprint (see Figure 6(*f*)).

5. Performance study

In the performance study conducted to assess the proposed approach, six scenes from four geographic locations were employed.[1] An objective evaluation technique is

proposed that estimates the performance indicators in terms of object-based, geometric, and pixel-based metrics. The details of the evaluation results have been discussed and analysed.

5.1. *Data sets*

Table 1 shows the characteristics of six test scenes from four data sets. As can be seen, the data sets have been chosen with varying point density from high to low. The Aitkenvale (AV) scene has five large buildings. The Hervey Bay (HB) scene contains many small buildings, for example, six were between 8 and 11 m^2 in area. The Vaihingen (VH) data set has three test scenes with slightly varying point density. There were also some small reference buildings in these scenes (two buildings were between 3 and 10 m^2 in VH 1; two between 4 and 15 m^2 in VH 2; and five between 8 and 15 m^2 in VH 3). In the Knox (KN) data set, there were the most number of small buildings among all test data sets: five buildings were between 3 and 10 m^2 and eight buildings were between 10 and 20 m^2 in area. Some buildings were not included in the reference set either due to absence of enough lidar points within the small building area or due to missing of the point cloud data.

For all data sets, 2D reference data were manually created from the orthoimagery. All visible buildings were digitized as polygons irrespective of their size. The reference data included garden sheds, garages, and so on. These were sometimes as small as 3 m^2 in area. However, due to occasional missing of points in lidar data or because of misalignment between the orthoimage and lidar data, there were some gaps in between each pair of reference and extracted boundaries. It can be exemplified when the extracted boundary is overlaid on the orthoimage for a building in the AV data set in Figure 6. In order to find the input point data P of a reference building, all the lidar points within the reference building were included in P. P is then given as the input to the proposed boundary extraction algorithm.

5.2. *Parameter setting*

Table 2 shows the parameters used by the proposed boundary extraction and regularization method.

The building boundary extraction algorithm in Section 3 uses a parameter $d_{\max}$, which is the maximum point-to-point distance. The value of this parameter depends on the input point cloud data. Different values have been set for the input data shown in

Table 1. Scenes in data sets.

Scenes	P_D (points m^{-2})	d_{max} (m)	N_B	N_C
AV	26.1	0.20	5	54
HB	12.3	0.50	22	161
VH 1	3.8	1.00	22	353
VH 2	4.2	1.00	11	216
VH 3	3.7	1.00	44	444
KN	1.1	1.20	50	430

AV, Aitkenvale; HB, Hervey Bay; VH, Vaihingen; KN, Knox. P_D = point density, $d_{\max}$ = maximum point-to-point distance, N_B = number of buildings, N_C = number of corners.

Table 2. Parameters used by the proposed roof extraction method.

Parameters	Values	Sources
Maximum point spacing d_{max}	See Table 1	Input lidar data
Point neighbourhood T_d	$2d_{max}$	Related to d_{max} (Sampath and Shan 2007)
Gaussian smoothing scale σ	3	Awrangjeb, Lu, and Fraser (2012)
Minimum length of a building side d_m	1 m	Awrangjeb and Fraser (2014b)
Minimum building width W_{min}	3 m	Awrangjeb, Ravanbakhsh, and Fraser (2010)
Minimum length of long lines	$2W_{min}$	Related to W_{min}
Angle threshold T_δ	$\frac{\pi}{16}$, $\frac{\pi}{4}$	This article

'This article' indicates that a corresponding parameter value has been chosen in this study.

Table 1. The neighbourhood threshold $T_d = 2d_{max}$ has been set in other studies (Sampath and Shan 2007; Dorninger and Pfeifer 2008; Awrangjeb and Fraser 2014b).

The regularization algorithm in Section 4 uses some additional parameters, including d_{max} and T_d. The corner and line extraction procedure in Section 4.1 uses the following extra parameters: The Gaussian smoothing parameter $\sigma = 3$ has been traditionally used in order to remove noise or irregularity (Awrangjeb, Lu, and Fraser 2012). The minimum length of a building side d_m is set at 1 m. Earlier, Awrangjeb and Fraser (2014b) set the dimension of the smallest building plane as 1 m. While detecting the missing corners, the horizontal distance threshold is considered to be the same as the smallest building side $d_m = 1$ m and vertical distance threshold to be d_{max}. These values are reasonable, since building sides smaller than these thresholds are hard to extract due to limitation of the input point cloud density. Moreover, small perpendicular building sides, if missed, are inserted as compensation in between extracted consecutive parallel lines.

The line adjustment and footprint generation procedure in Section 4.2 uses the following extra parameters: While for large buildings, the principal directions are estimated based on the lines longer than twice the minimum building width W_{min}, shorter lines are considered for small buildings. $W_{min} = 3$ m has been used widely in the literature (Awrangjeb, Ravanbakhsh, and Fraser 2010). Earlier, Sampath and Shan (2007) used the minimum length of a long line as 10 m. The use of 6 m by the proposed method is reasonable because there may be many buildings where a building side may not be as long as 10 m. For example, Figure 9(*a*) shows that the length of the longest side of the majority (more than 65%) of the buildings in the test data sets is less than 6 m. This threshold is used to have additional principal directions that may present in large buildings. However, there are many small buildings in the test data sets and about 30% of the buildings have a length smaller than 3 m (see Figure 9(*a*)). Thus, as a precaution, if a building does not have a 6 m long side, the proposed method iteratively decreases the threshold to find a long side that is shorter than 6 m. This means it is not necessary to set a low threshold for large buildings, which may cause a wrong estimation of the principal direction for large buildings.

For regularization, two values are considered for the angle threshold T_δ: $\frac{\pi}{16}$ for long lines and $\frac{\pi}{4}$ for short lines. $T_\delta = \frac{\pi}{16}$ or 11.25° is used to decide the principal direction of a building once its long sides are obtained. $T_\delta = \frac{\pi}{4}$ or 45° is used to adjust the short lines with respect to the principal direction. In order to test the sensitivity of T_δ, eight values were tested for long and short lines, as shown in the horizontal axis of Figure 9(*b*). There was no or negligible performance difference observed in terms of (both object- and pixel-based) completeness, correctness, and quality. Figure 9(*b*) shows the rotation angle

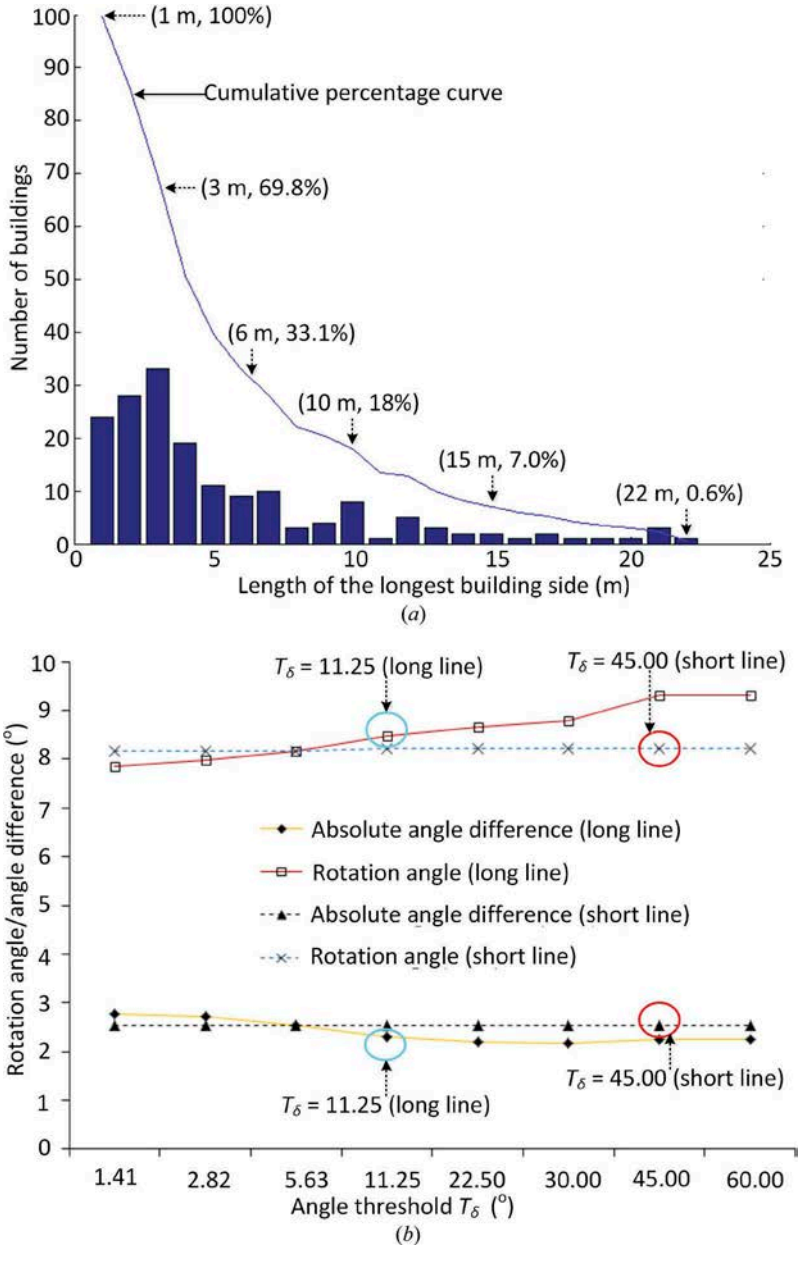

Figure 9. Analysis for parameter setting: (*a*) number of buildings plotted against the length of the longest building side (e.g. 33.1% buildings have a side of 6 m or longer), and (*b*) rotation angle δ_a used to adjust lines and absolute angle difference γ_a after adjustment plotted against the angle threshold T_δ for short and long lines.

δ_a to adjust the lines to the principal direction and the absolute angle difference γ_a with the reference building sides after the adjustment. As can be seen, these two performance indicators also did not change much for different T_δ values. Consequently, any values of T_δ can be chosen for long and short lines. The selection of $T_\delta = 11.25^\circ$ allows to

determine four principal directions (at multiple of $\frac{\pi}{16}$). Since the extracted short lines can be oriented randomly, $T_\delta = 45^\circ$ for short lines is quite acceptable given the low input point density, especially in the KN data set.

5.3. *Evaluation system and indices*

Evaluation systems (e.g. Rutzinger, Rottensteiner, and Pfeifer 2009) that involve one or more overlap thresholds for deciding whether an extracted boundary is correct or not may be biased. The use of such overlap thresholds does not work well in all cases, and this approach can be controversial since there is no unique way to select these thresholds (Shufelt 1999). Moreover, the estimated performance may be drastically affected when the area of the extracted building boundary changes (Awrangjeb and Fraser 2014a). Consequently, the threshold-free evaluation system in Awrangjeb and Fraser (2014a) has been employed in this study.

For geometric performance measurements, Rutzinger, Rottensteiner, and Pfeifer (2009) did not present any evaluation metrics, except the root mean square error (RMSE), that can robustly estimate the geometric accuracy of the extracted line and footprint. Zeng, Wang, and Lehrbass (2013) proposed a single overall metric through combining individual metrics. A single metric can make the comparison of different building extraction techniques straightforward. However, the derivation of such a single metric is subjective because it requires setting of weights to different metrics. Nevertheless, these weights depend on applications. In addition, a single evaluation metric fails to detail the performance of a building extraction technique. Thus, in this study, in addition to using the popular measurements such as completeness, correctness, quality, and the RMSE to measure the quality of the extracted boundary and footprint, the absolute perpendicular distance, rotation angle, and the absolute angle difference have been proposed to rigorously estimate the extracted line and footprint. Detailed reviews of other evaluation systems and performance metrics can be found in Zeng, Wang, and Lehrbass (2013) and Awrangjeb and Fraser (2014a).

In this study, the evaluation is done in two steps: first, for extracted irregular boundaries and regularized lines, and second, for generated building footprints. Each extracted irregular building boundary is evaluated with respect to its reference boundary using an area- or pixel-based evaluation. A grid G of pixels at 0.5 m resolution is generated to estimate completeness C_{mb}, correctness C_{rb}, quality Q_{lb}, area omission error A_{oeb}, and area commission error A_{ceb} (Awrangjeb and Fraser 2014a). The regularized lines are evaluated in terms of the goodness of fit with respect to the extracted boundary. The mean of the absolute perpendicular distance d_{p} from the points of the corresponding smoothed curve segment to a line is used. In addition, the absolute angle δ_{a} at which the extracted line is rotated in order to make it parallel or perpendicular to a principal direction is measured. The lower values for d_{p} and δ_{a} indicate a better fit of the extracted line.

To evaluate the generated building footprints, each extracted footprint $\mathrm{B_d}$ is then compared with its corresponding reference footprint $\mathrm{B_r}$. Let the set of corner points in $\mathrm{B_d}$ be $\{\mathrm{C_{d1}}, \mathrm{C_{d2}}, \ldots, \mathrm{C_{dm}}\}$ and those in $\mathrm{B_r}$ be $\{\mathrm{C_{r1}}, \mathrm{C_{r2}}, \ldots, \mathrm{C_{rn}}\}$, where m and n are the number of corners in $\mathrm{B_d}$ and $\mathrm{B_r}$, respectively. Starting from the closest pair of extracted and reference corner points $(\mathrm{C_{d}}_i, \mathrm{C_{r}}_j)$, where $1 \leq i \leq m$ and $1 \leq j \leq n$, it is

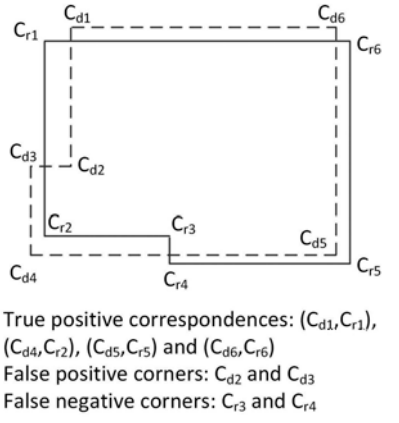

Figure 10. Making correspondences between a reference (solid polygon) and an extracted (dashed polygon) footprints.

checked whether the two corresponding lines in B_d are parallel to two corresponding lines in B_r. If they are parallel (angle $\gamma \leq \frac{\pi}{4}$), (C_{di}, C_{rj}) is a true positive (TP) corner pair. If they are not, the next closest corner pair is ascertained in the same way. Once two or more TP corner pairs are marked, marking of a successive corner pair as a TP is subject to a further condition, where the new TP corner pair must not violate the corner topology. This latter check stops corner pairs being marked as TP inappropriately. For example, in Figure 10, after corner pairs (C_{d5}, C_{r5}), (C_{d6}, C_{r6}), (C_{d4}, C_{r2}), and (C_{d1}, C_{r1}) have been marked as TP, the next closest corner pair (C_{d2}, C_{r3}) cannot be marked as a TP, because C_{d2} and C_{r3} are in between different TP corner pairs. Consequently, C_{d2} and C_{d3} are marked as false positive (FP) and C_{r3} and C_{r4} are marked as false negative (FN).

Once the corner correspondences are established between the extracted and reference footprints, object-based, geometric, and pixel-based evaluation metrics are estimated as follows. In object-based evaluation, completeness C_m, correctness C_r, and quality Q_l metrics are measured using the number of TP, FP, and FN corners. In geometric evaluation, the RMSE is determined for TP corner correspondences. The absolute angle difference γ_a between the parallel lines of TP corner pairs indicates the angular difference between a reference corner and its corresponding extracted corner. This angular estimation directly represents the orientation accuracy for the estimated principal directions. In pixel-based evaluation, the same grid G is used to estimate completeness C_{mp}, correctness C_{rp}, quality Q_{lp}, area commission error A_{ce}, and area omission error A_{oe} for the generated footprint.

5.4. *Evaluation results and discussions*

The evaluation results for the extracted boundary and line are first presented. Then, the generated building boundary is compared with the reference boundary using the proposed corner-based objective evaluation system. Some visual results of generated building footprints by the proposed method are provided. Finally, the performance of

the proposed boundary and line extraction is compared with those provided by the α-shape (Edelsbrunner, Kirkpatrick, and Seidel 1983) and DP (Douglas and Peucker 1973) algorithms.

5.4.1. *Extracted boundary and line*

Table 3 shows the area- or pixel-based evaluation of the extracted building boundaries for the test scenes. It also presents the goodness of fit of the regularized lines with respect to the extracted boundary.

For the extracted boundaries, the correctness values were almost 100% because the input point sets were exactly within the reference building boundaries. The observation is also supported by close to zero area commission errors for all test data sets. However, the completeness values were smaller than the correctness values because pixels along the reference building footprints were not in the extracted building boundaries due to low point density. They decreased with the decrease of input point density. Although the best pixel-based completeness was found for the AV data set due to its high point density, the worst performance was observed in the KN data set since the extracted boundaries were determined much smaller than the original size due to its low point density. This fact is also evident from the lowest and highest area omission errors for the AV and KN data sets, respectively.

For the regularized extracted lines, the mean of the absolute perpendicular distance d_p is the lowest for the AV data set since the input point cloud is the densest. Compared to other test data sets, d_p is high for the VH data set, because some buildings in these data sets have complex shapes and successive parallel building sides may be merged due to low point density. In terms of the rotation angle δ_a, the KN data set shows the worst result as the estimation of the principal direction may be mistaken in some cases, especially for small buildings, which causes the extracted lines to be rotated at large angles. In contrast, δ_a is the lowest for the HB data set because here a large number of buildings are of simple rectangular shape. As can be seen in the histograms of δ in Figures 11(*a*) and (*b*), while a large number of lines are rotated by large angles in the KN data set, the majority of the lines are rotated by small angles for the HB data set.

5.4.2. *Generated footprint*

Table 4 shows the objective evaluation results for the generated building footprints. Although the AV data set has the highest input point density, its object-based

Table 3. Evaluation of the extracted boundary and line.

Scenes	C_{mb} (%)	C_{rb} (%)	Q_{lb} (%)	A_{oeb} (%)	A_{ceb} (%)	d_p (m)	δ_a (°)
AV	96.6	100	96.2	3.4	0	0.038	11.23
HB	94.1	100	93.7	5.9	0	0.049	4.67
VH 1	92.4	99.5	92.0	7.6	0.6	0.077	8.26
VH 2	92.6	99.6	92.3	7.4	0.4	0.081	6.50
VH 3	89.2	99.7	88.6	10.9	0.3	0.079	7.41
KN	75.5	99.6	74.7	24.5	0.4	0.075	13.89
Average	90.1	99.7	89.6	10.0	0.3	0.067	8.66

AV, Aitkenvale; HB, Hervey Bay; VH, Vaihingen; KN, Knox. Indices for pixel-based evaluation of boundary are C_{mb} = completeness, C_{rb} = correctness, Q_{lb} = quality, A_{oeb} = area omission error, and A_{ceb} = area commission error. Indices for geometric evaluation of line are d_p = mean of the absolute perpendicular distance and δ_a = rotation angle to fit the line with the principal direction.

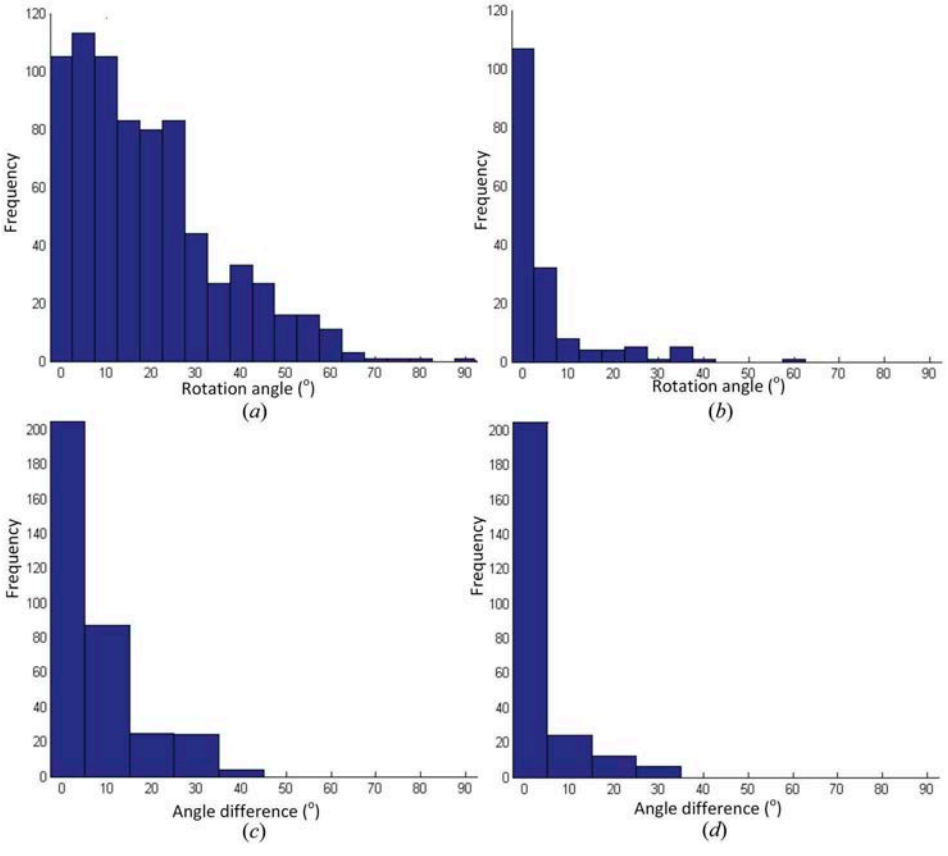

Figure 11. Histograms: rotation angles used to adjust the extracted lines with principal directions for (*a*) Knox and (*b*) Hervey Bay data sets. Absolute angle difference between extracted and reference lines for true positive correspondences for (*c*) Knox and (*d*) Vaihingen Scene 1 data sets.

Table 4. Objective evaluation for building footprint.

Scenes	C_m	C_r	Q_l	C_{mp}	C_{rp}	Q_{lp}	A_{oe}	A_{ce}	RMSE (m)	γ_a (°)
AV	89.5	85.0	74.5	95.2	99.4	94.8	4.8	0.6	0.51	0.41
HB	97.0	89.4	86.9	92.3	99.4	92.0	7.7	0.6	0.31	1.27
VH 1	73.5	82.8	62.3	90.4	98.0	90.0	9.6	2.0	0.65	2.73
VH 2	89.8	81.2	74.1	91.7	99.1	91.5	8.3	0.9	0.73	3.36
VH 3	84.2	81.8	69.1	86.9	99.1	86.4	13.1	0.9	0.64	1.89
KN	89.7	73.9	67.5	72.3	99.2	71.5	27.7	0.8	0.87	4.90
Average	87.3	82.3	72.4	88.1	99.0	87.7	11.9	1.0	0.62	2.42

AV, Aitkenvale; HB, Hervey Bay; VH, Vaihingen; KN, Knox. Indices for object-based evaluation of corner are C_m = completeness, C_r = correctness, and Q_l = quality (all in percentage). Indices for pixel-based evaluation of footprint are C_{mp} = completeness, C_{rp} = correctness, Q_{lp} = quality, A_{oe} = area omission error, and A_{ce} = area commission error (all in percentage). Indices for geometric evaluation of footprint are RMSE = root mean square error (for corner) and γ_a = absolute angle difference (for line).

performance was worse than that in the HB data set, which exhibited the best performance (Figure 12(*a*)). There were two reasons. First, in the AV data set, there were many small sides, many of which were merged with the neighbouring long building sides. Second, some spurious corners were detected in the AV data set due to noisy or missing

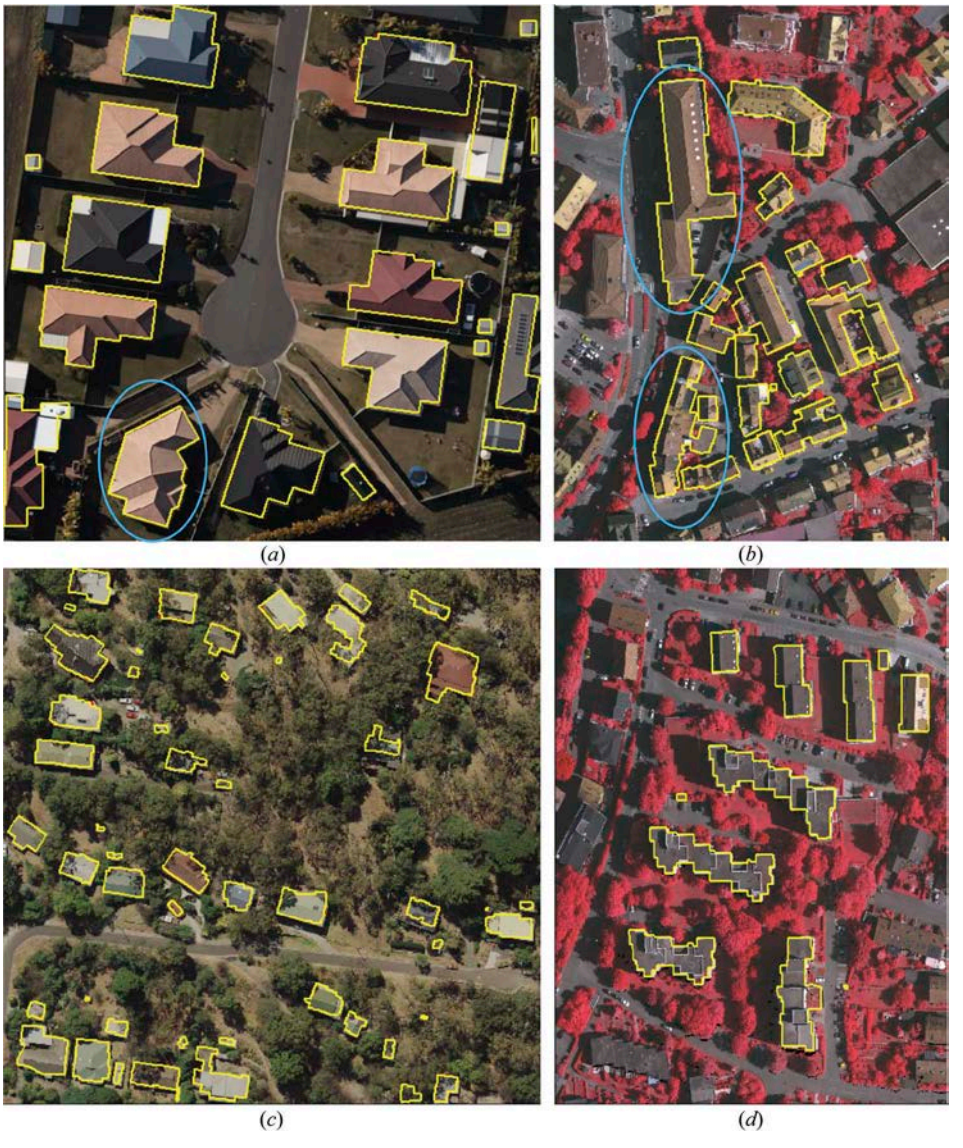

Figure 12. Extracted building footprints (yellow polygons) for data sets: (*a*) Hervey Bay, (*b*) Vaihingen Scene 1, (*c*) Knox, and (*d*) Vaihingen Scene 2. The cyan ellipses indicate some examples where the proposed method was capable of extracting correct footprints for buildings with more than one principal direction.

point data. Among the three scenes of the VH data set, the best object-based performance was observed in the VH 2 scene because of its regular rectilinear building structure (Figure 12(*d*)). The worst performance was obtained for the KN data set (Figure 12(*c*)). Many false-positive corners were detected due to its low point density (1.2 points m^{-2}).

The area- or pixel-based evaluation results for the generated building footprints in Table 4 are similar to those for the extracted boundaries in Table 3, but with a slight

degradation. This is an indication that the proposed regularization procedure offers building footprints resembling to the extracted boundaries.

In geometric evaluation, RMSE values were within one to two times of the maximum point-to-point distance d_{max} in the input data. Again, the largest error was found for the KN data set due to its low point density. This performance trend was also supported by the angle difference γ_a between the extracted and reference footprint lines for true correspondences. γ_a directly indicates the accuracy of the estimated principal directions of buildings. While for the AV data set γ_a value was the lowest, for the KN data set its value was the highest. The histogram in Figure 11(*c*) shows that many footprint lines in the KN data set had high angle differences (10°–40°). In contrast, the histogram in Figure 11(*d*) shows that only a few footprint lines in Scene 1 of the VH data set had high angle differences (10°–30°).

5.4.3. *Visual performance*

The cyan colour ellipses in Figures 12(*a*) and (*b*) show that the proposed footprint extraction procedure is capable of extracting correct footprint for buildings with more than one principal direction. The white colour ellipses in Figures 13(*a*)–(*f*) show that it is also capable of extracting small details in the building footprints. However, the red colour ellipses in Figure 13(*c*) indicate that when the small details are not parallel or perpendicular to the principal directions or if the point density is low, it is unable to extract small details along the building boundary. Figures 13(*g*)–(*j*) show some extracted footprints where some of the principal directions were incorrectly estimated due to low point density in the KN data set. Consequently, the orientation of some part (Figure 13 (*g*)) or the whole (Figures 13(*h*)–(*j*)) of extracted footprints (yellow polygons) was wrong with respect to reference footprints (cyan polygons). Although the corner topology was

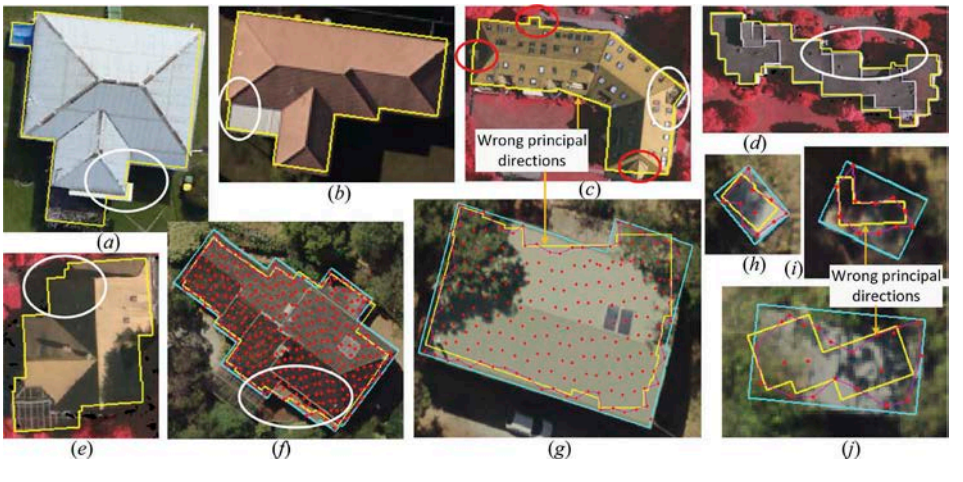

Figure 13. Some examples of extracted footprints taken from (*a*) Aitkenvale, (*b*) Hervey Bay, (*c*) Vaihingen Scene 1, (*d*) Vaihingen Scene 2, (*e*) Vaihingen Scene 3, and (*f–j*) Knox scenes. Yellow polygon = extracted footprint, cyan polygon = reference footprint, magenta polygon = extracted boundary, and red dots = input points.

correct for the extracted footprint in Figure 13(*h*), it was incorrect for the extracted footprints in Figures 13(*i*)–(*j*) due to low point density.

5.4.4. *Comparative results*

The study by Zhang, Yan, and Chen (2006) does not present any boundary extraction method. However, it presents a boundary regularization method, in which the adjustment of the edge segments with respect to a single dominant direction does not work when building boundary parts are aligned in a different direction, as highlighted in Figures 12(*a*) and (*b*). In order to alleviate this problem, the method suggests for a manual refinement step for the oblique edge segments. The boundary extraction technique by Sampath and Shan (2007) fits a rectangular neighbourhood with respect to the scan direction of the point cloud. This means for a given point cloud when the information related to the scan direction is not available, the rectangular neighbourhood becomes a circular neighbourhood used in its predecessor (Jarvis 1977). Moreover, similar to other convex hull algorithms, this technique fails to extract any holes that may present inside a shape.

Thus, in order to compare the proposed boundary and line extraction procedures, the α-shape (Edelsbrunner, Kirkpatrick, and Seidel 1983) as well the DP (Douglas and Peucker 1973) algorithms were implemented. These are two popular methods which have been expensively used by the scientist and researcher in different applications. The value of α was set at $d_{\max}$ and the height threshold of the DP algorithm was set at $\frac{1}{\sqrt{2}}$ considering the minimum building side $d_{\mathrm{m}} = 1$ m. The α-shapes were first generated and then the DP lines were extracted for the extracted shapes. For building footprint generation, the proposed procedure in Section 4.2 was employed.

Table 5 shows the pixel-based evaluation of the extracted building boundaries by the α-shape algorithm. It also presents the goodness of fit of the DP lines (Douglas and Peucker 1973) from the extracted α-shape. Comparing these results with those in Table 3, it is evident that the proposed boundary extraction algorithm showed better pixel-based completeness with lower area omission error, although the correctness and area commission error were similar. The extracted lines by the proposed line extraction algorithm had not only lower distance error for all test scenes, but also lower rotation angle to fit lines, except for the AV scene for which the proposed technique offered slightly higher rotation error. Note that the worse average results in Table 5 are highlighted in italic font.

Table 6 shows the objective evaluation results for the generated building footprints when the α-shape and the DP line were used with the footprint generation step in Section 4.2.

Table 5. Evaluation of the extracted boundary and line (using α-shape and Douglas–Peucker line).

Scenes	C_{mb} (%)	C_{rb} (%)	Q_{lb} (%)	A_{oeb} (%)	A_{ceb} (%)	d_{p} (m)	δ_{a} (°)
AV	96.4	100	96.1	3.6	0	0.101	10.57
HB	92.6	100	92.3	7.4	0	0.107	4.79
VH 1	90.9	99.6	90.4	9.1	0.4	0.098	9.02
VH 2	91.7	99.7	90.8	8.3	0.3	0.101	8.79
VH 3	87.4	99.8	86.6	12.6	0.2	0.103	7.45
KN	72.1	99.8	71.0	27.9	0.2	0.097	14.23
Average	*88.5*	99.8	*87.9*	*11.5*	0.2	*0.101*	*9.14*

AV, Aitkenvale; HB, Hervey Bay; VH, Vaihingen; KN, Knox. Indices for pixel-based of boundary are C_{mb} = completeness, C_{rb} = correctness, Q_{lb} = quality, A_{oeb} = area omission error, and A_{ceb} = area commission error. Indices for geometric evaluation of line are d_{p} = mean of the absolute perpendicular distance and δ_{a} = rotation angle to fit the line with the principal direction.

Table 6. Objective evaluation for building footprint (using α-shape and Douglas–Peucker line).

Scenes	C_m	C_r	Q_l	C_{mp}	C_{rp}	Q_{lp}	A_{oe}	A_{ce}	RMSE (m)	γ_a (°)
AV	85.5	86.4	71.9	95.5	99.4	95.4	4.5	0.6	0.47	1.84
HB	94.7	89.9	86.0	89.3	98.0	88.7	10.7	2.0	0.39	1.80
VH 1	74.5	78.3	61.1	90.0	99.2	89.7	10.1	0.8	0.64	3.10
VH 2	91.8	72.1	66.9	91.1	99.3	91.0	8.9	0.8	0.63	4.84
VH 3	86.2	74.9	65.4	86.7	99.2	86.0	13.3	0.8	0.61	2.88
KN	93.8	71.2	67.5	71.4	99.6	70.6	28.6	0.4	0.88	5.50
Average	87.8	*78.8*	*69.8*	*87.3*	99.1	*86.9*	*12.7*	0.9	0.60	*3.33*

AV, Aitkenvale; HB, Hervey Bay; VH, Vaihingen; KN, Knox. Indices for object-based evaluation of corner are C_m = completeness, C_r = correctness, and Q_l = quality (all in percentage). Indices for pixel-based of footprint are C_{mp} = completeness, C_{rp} = correctness, Q_{lp} = quality, A_{oe} = area omission error, and A_{ce} = area commission error (all in percentage). Indices for geometric evaluation of footprint are RMSE = root mean square error (for corner) and γ_a = absolute angle difference (for line).

Comparing with Table 4, it is observed that the footprints using the proposed boundary and line extraction procedures offered higher object- and pixel-based qualities, and lower area omission error and angle difference. However, they offered slightly lower object-based completeness and pixel-based correctness for some test scenes. Note that the worse average results in Table 6 are highlighted in italic font.

Some examples of boundary and line extraction and footprint generation are shown in Figures 14 and 15. In Figure 14, only boundary generated by the proposed technique is shown, since in high-density point cloud it is hard to visualize difference with the one generated by the α-shape algorithm. However, for low-density point cloud it is seen that the α-shape algorithm misses some true building area (Figure 15(*a*), within orange circle), while the proposed technique did not (Figure 15(*d*)). While the DP algorithm either missed some lines (Figure 14(*c*), within cyan ellipse) or extracted many unnecessary lines (Figure 15(*b*)), the proposed line extraction technique obtained better results (Figures 14(*d*) and 15(*e*)). As a consequence, the line adjustment and footprint generation technique proposed in Section 4.2 found better footprints for the proposed boundary and line extraction techniques than the combination of the α-shape and DP algorithms (Figures 14(*e*) vs (*f*) and Figures 15(*c*) and (*f*)).

6. Conclusion

A detailed solution for the extraction of building outline from the input point cloud data has been proposed. The solution has essentially three major steps: boundary identification, tracing, and regularization. The initial boundary (in terms of point edges) from the Delaunay triangulation of the input point set is gradually refined to determine the building boundary. The boundary of an inside 'hole' or 'cavity' can also be determined by using a similar approach. An extracted boundary is traced to generate a series of points. For a boundary that consists of two or more loops is decomposed into segments of edges, which are latter merged into a single series of points. The regularization step extracts corners and lines from the irregular boundary and obtains a polygon representing a rectilinear building footprint. An evaluation system based on the number of corners is proposed and employed to validate the effectiveness of the proposed building outline generation technique.

The benefits of the proposed solutions are as follows: First, for a given point cloud data, the proposed boundary extraction technique can extract outside boundary as well

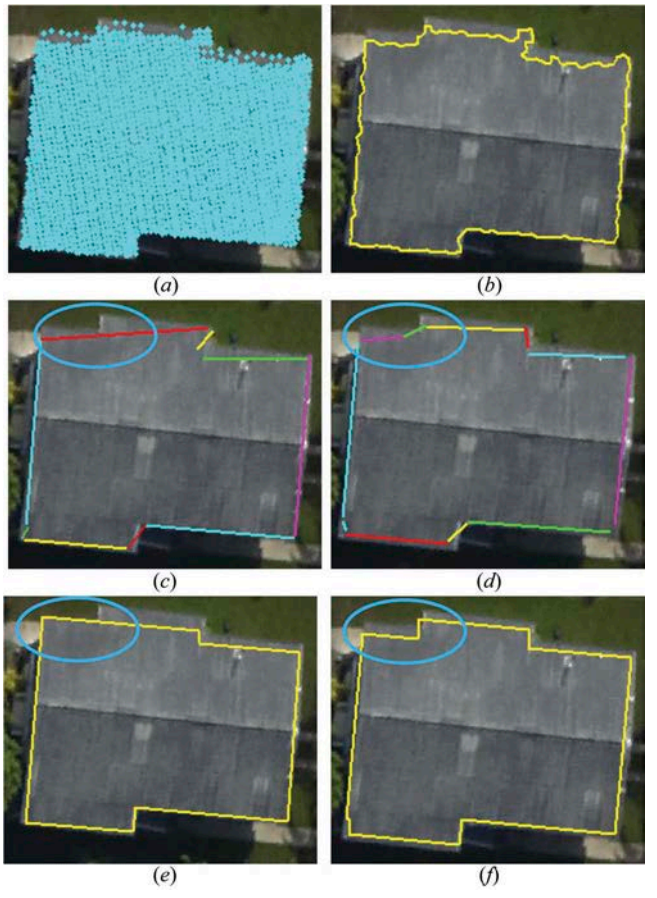

Figure 14. Examples taken from Aitkenvale scene: (*a*) input point cloud, (*b*) extracted boundary, (*c*) lines by Douglas–Peucker algorithm, (*d*) lines by proposed technique and (*e*, *f*) footprints from (*c*) and (*d*), respectively, using proposed technique in Section 4.2.

as the boundary of any inside 'hole' or 'cavity' present in the data. Second, an important property of the proposed boundary extraction algorithm is that it can extract the boundary of a large point set from the boundaries of two or more of its point subsets. Any inside hole can also be determined from the boundaries of subsets. This is particularly useful for obtaining a building boundary from the boundaries of its planes. Third, in general, contour-based corner detectors are not applicable to contours extracted from irregular point cloud. The proposed improved corner and line extraction technique is capable of detecting corners and extracting lines from the irregular boundary. Finally, experimental results have shown that the proposed boundary and line extraction techniques offer better performance in terms of pixel-based completeness and geometric accuracy (by comparing Table 3 with Table 5). They can preserve detail along the building boundary, even in low-density input data.

In practice, there may be two or more buildings of different heights but with shared boundary edges. There may also be a building, for example, a high-rise building in the city, which has different heights at different parts. Since the proposed boundary extraction technique considers only 2D boundary points (excluding the height value), it would

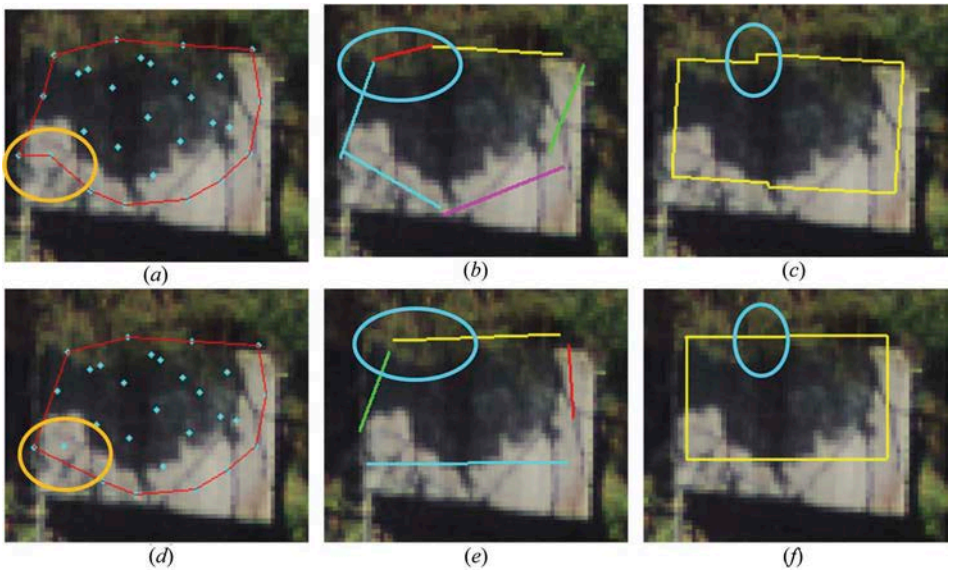

Figure 15. Boundary and lines taken from Knox scene: (*a*) *α*-shape, (*b*) Douglas–Peucker lines from (*a*), (*c*) footprint from (*b*), (*d*–*f*) boundary, lines, and footprint, respectively, produced using the proposed methods. In both cases, footprints were generated using proposed technique in Section 4.2.

extract a single boundary in such scenarios. In addition, the proposed regularization technique sometimes fails to properly determine the principal direction, especially when the input point density is low as exemplified in the Knox data set. The future work will focus on the extraction of boundaries for buildings with different heights but with shared boundary edges, the more appropriate alignment of the buildings with low point density and generation of three-dimensional building models.

Note

1. Available at http://users.monash.edu.au/mawrangj/RExtraction.html.

Acknowledgements

Dr. Awrangjeb is the recipient of the Discovery Early Career Researcher Award by the Australian Research Council (project number DE120101778). The Vaihingen data set was provided by the German Society for Photogrammetry, Remote Sensing and Geoinformation (DGPF) (Cramer 2010). The Aitkenvale and Hervey Bay data sets were provided by Ergon Energy in Queensland, Australia. The Knox data set was provided by the Department of Environment and Primary Industries of Victoria, Australia.

Disclosure statement

No potential conflict of interest was reported by the author.

Funding

This work was supported by the Australian Research Council [grant number DE120101778].

References

Alharthy, A., and J. Bethel. 2002. "Heuristic Filtering and 3D Feature Extraction from LIDAR Data." *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XXXIV (Part 3A): 29–34.

Awrangjeb, M., and C. S. Fraser. 2014a. "An Automatic and Threshold-Free Performance Evaluation System for Building Extraction Techniques from Airborne LIDAR Data." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 7 (10): 4184–4198. doi:10.1109/JSTARS.2014.2318694.

Awrangjeb, M., and C. S. Fraser. 2014b. "Automatic Segmentation of Raw LIDAR Data for Extraction of Building Roofs." *Remote Sensing* 6 (5): 3716–3751. doi:10.3390/rs6053716.

Awrangjeb, M., G. Lu, and C. S. Fraser. 2012. "Performance Comparisons of Contour-Based Corner Detectors." *IEEE Transactions on Image Processing* 21 (9): 4167–4179. doi:10.1109/TIP.2012.2200493.

Awrangjeb, M., G. Lu, and C. S. Fraser. 2014. "Automatic Building Extraction from LIDAR Data Covering Complex Urban Scenes." *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XL (Part 3): 25–32. doi:10.5194/isprsarchives-XL-3-25-2014.

Awrangjeb, M., M. Ravanbakhsh, and C. S. Fraser. 2010. "Automatic Detection of Residential Buildings Using LIDAR Data and Multispectral Imagery." *ISPRS Journal of Photogrammetry and Remote Sensing* 65 (5): 457–467. doi:10.1016/j.isprsjprs.2010.06.001.

Cramer, M. 2010. "The DGPF-Test on Digital Airborne Camera Evaluation – Overview and Test Design." *Photogrammetrie "Fernerkundung" Geoinformation* 2010: 73–82. doi:10.1127/1432-8364/2010/0041.

De Berg, M., M. Van Kreveld, M. Overmans, and O. Schwarzkopf. 2000. *Computational Geometry: Algorithms and Applications*. 2nd ed. Berlin: Springer-Verlag.

Dorninger, P., and N. Pfeifer. 2008. "A Comprehensive Automated 3D Approach for Building Extraction, Reconstruction, and Regularization from Airborne Laser Scanning Point Clouds." *Sensors* 8 (11): 7323–7343. doi:10.3390/s8117323.

Douglas, D., and T. Peucker. 1973. "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature." *Cartographica*: *The International Journal for Geographic Information and Geovisualization* 10 (2): 112–122. doi:10.3138/FM57-6770-U75U-7727.

Edelsbrunner, H., D. G. Kirkpatrick, and R. Seidel. 1983. "On the Shape of a Set of Points in the Plane." *IEEE Transactions on Information Theory* 29: 551–559. doi:10.1109/TIT.1983.1056714.

Galvanin, E. A. S., and A. P. Dal Poz. 2012. "Extraction of Building Roof Contours from LIDAR Data Using a Markov-Random-Field-Based Approach." *IEEE Transactions on Geoscience and Remote Sensing* 50 (3): 981–987. doi:10.1109/TGRS.2011.2163823.

Grigillo, D., and U. Kanjir. 2012. "Urban Object Extraction from Digital Surface Model and Digital Aerial Images." *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* I (Part 3): 215–220. doi:10.5194/isprsannals-I-3-215-2012.

Haithcoat, T. L., W. Song, and J. D. Hipple. 2001. "Building Footprint Extraction and 3-D Reconstruction from LIDAR Data." In *IEEE/ISPRS Joint Workshop on Remote Sensing and Data Fusion over Urban Areas*, 74–78. Rome: IEEE. http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7736

Hinks, T., H. Carr, D. F. Laefer, Y. Morvan, C. O'Sullivan, L. Truong-Hong, and S. Ceribasi. 2009. "Robust Building Outline Extraction." Patent PTO 56793223.

Jarvis, R. A. 1977. "Computing the Shape Hull of Points in the Plane." In *IEEE Computer Society Conference Pattern Recognition and Image Processing*, New York, 231–241.

Jwa, Y., G. Sohn, V. Tao, and W. Cho. 2008. "An Implicit Geometric Regularization of 3D Building Shape Using Airborne LIDAR Data." *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XXXVI (Part 5): 69–76.

Kwak, E., and A. Habib. 2014. "Automatic Representation and Reconstruction of DBM from LIDAR Data Using Recursive Minimum Bounding Rectangle." *ISPRS Journal of Photogrammetry and Remote Sensing* 93 (7): 171–191. doi:10.1016/j.isprsjprs.2013.10.003.

Laefer, D. F., T. Hinks, H. Carr, and L. Truong-Hong. 2011. "New Advances in Automated Urban Modelling from Airborne Laser Scanning Data." *Recent Patents in Engineering* 5 (3): 196–208. doi:10.2174/187221211797636890.

Lahamy, H. 2008. "Outlining buildings using airborne laser scanner data." Ph.D. thesis. www.itc.nl/library/papers_2008/msc/gfm/lahamy.pdf

Maas, H., and G. Vosselman. 1999. "Two Algorithms for Extracting Building Models from Raw Laser Altimetry Data." *ISPRS Journal of Photogrammetry and Remote Sensing* 54 (2–3): 153–163. doi:10.1016/S0924-2716(99)00004-0.

Peethambaran, J., and R. Muthuganapathy. 2015. "A Non-Parametric Approach to Shape Reconstruction from Planar Point Sets through Delaunay Filtering." *Computer-Aided Design* 62 (5): 164–175. doi:10.1016/j.cad.2014.12.002.

Perera, S. N., H. A. Nalani, and H.-G. Maas. 2012. "An Automated Method for 3D Roof Outline Generation and Regularization in Airborne Laser Scanner Data." *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* I (Part 3): 281–286.

Pu, S., and G. Vosselman. 2007. "Extracting Windows from Terrestrial Laser Scanning." In *ISPRS Workshop on Laser Scanning and SilviLaser 2007*, edited by P. Rönnholm, H. Hyyppä, and J. Hyyppä, 320–325. Espoo, September. http://www.isprs.org/proceedings/XXXVI/3-W52/

Rutzinger, M., F. Rottensteiner, and N. Pfeifer. 2009. "A Comparison of Evaluation Techniques for Building Extraction from Airborne Laser Scanning." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 2 (1): 11–20. doi:10.1109/JSTARS.2009.2012488.

Sampath, A., and J. Shan. 2007. "Building Boundary Tracing and Regularization from Airborne LIDAR Point Clouds." *Photogrammetric Engineering & Remote Sensing* 73 (7): 805–812. doi:10.14358/PERS.73.7.805.

Shufelt, J. 1999. "Performance Evaluation and Analysis of Monocular Building Extraction from Aerial Imagery." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21 (4): 311–326. doi:10.1109/34.761262.

Truong-Hong, L., D. Laefer, T. Hinks, and H. Carr. 2012. "Flying Voxel Method with Delaunay Triangulation Criterion for Façade/Feature Detection for Computation." *Journal of Computing in Civil Engineering* 26 (6): 691–707. doi:10.1061/(ASCE)CP.1943-5487.0000188.

Verma, V., R. Kumar, and S. Hsu. 2006. "3D Building Detection and Modeling from Aerial LIDAR Data." In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, edited by A. Fitzgibbon, C. J. Taylor, and Y. LeCun, 22213–22220. New York: IEEE. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=1641024&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D1641024

Wang, J., and J. Shan. 2009. "Segmentation of LIDAR Point Clouds for Building Extraction." In *ASPRS Annual Conference*, 9–13. Baltimore, MD: ASPRS.

Weidner, U., and W. Förstner. 1995. "Towards Automatic Building Extraction from High-Resolution Digital Elevation Models." *ISPRS Journal of Photogrammetry and Remote Sensing* 50 (4): 38–49. doi:10.1016/0924-2716(95)98236-S.

Yang, B., W. Xu, and Z. Dong. 2013. "Automated Extraction of Building Outlines from Airborne Laser Scanning Point Clouds." *IEEE Geoscience and Remote Sensing Letters* 10 (6): 1399–1403. doi:10.1109/LGRS.2013.2258887.

Zeng, C., J. Wang, and B. Lehrbass. 2013. "An Evaluation System for Building Footprint Extraction from Remotely Sensed Data." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 6 (3): 1640–1652. doi:10.1109/JSTARS.2013.2256882.

Zhang, K., J. Yan, and S.-C. Chen. 2006. "Automatic Construction of Building Footprints from Airborne LIDAR Data." *IEEE Transactions on Geoscience and Remote Sensing* 44 (9): 2523–2533. doi:10.1109/TGRS.2006.874137.

Zhou, Q.-Y., and U. Neumann. 2013. "Complete Residential Urban Area Reconstruction from Dense Aerial LIDAR Point Clouds." *Graphical Models* 75 (3): 118–125. doi:10.1016/j.gmod.2012.09.001.